PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 15/00 | A2

(11) International Publication Number: WO 99/26159

(43) International Publication Date: 27 May 1999 (27.05.99)

(21) International Application Number: PCT/US98/23347

(22) International Filing Date: 2 November 1998 (02.11.98)

(30) Priority Data:
60/065,991 14 November 1997 (14.11.97) US
09/093,969 8 June 1998 (08.06.98) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: PARSONS, John, E.; 2406 222nd Avenue N.E., Redmond, WA 98053 (US). GRAZIADIO, Bradley, J.; 2827 233rd Place N.E., Redmond, WA 98053 (US). MO-MOH, Oshoma; 1602 E. Garfield Avenue #C, Seattle, WA 98112 (US).

(74) Agents: LEE, Lewis, C. et al.; Suite 430, W. 201 North River Drive, Spokane, WA 99201 (US).

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SERVER OPERATING SYSTEM FOR SUPPORTING MULTIPLE CLIENT-SERVER SESSIONS AND DYNAMIC RECONNECTION OF USERS TO PREVIOUS SESSIONS

(57) Abstract

A server operating system supports multiple client-server sessions and enables a user to begin a session and later dynamically reconnect to that session even if the user uses two different client computers. The operating system has a multi-user session manager to enable multiple client-server sessions on the server and a multi-user stack protocol manager to manage one or more protocol stacks used in communicating with the clients. When a user connects to the server via a first client, the stack protocol manager assigns a first protocol stack to this first client-server connection and the session manager creates a first session for the user. When the user subsequently reconnects to the server using a second client that is different from the first client, the stack manager assigns a second protocol stack to a second client-server connection and the session begins creating a second session for the user. During this latter process, however, the session manager recognizes that the user is affiliated with the first session. The session manager adapts the first session to conform to the system configuration of the second client. The session manager then reassociates the second protocol stack with the reconfigured first session so that the user is returned to his/her original session, even though they logged on from a different client.

SERVER OPERATING SYSTEM FOR SUPPORTING MULTIPLE CLIENT-SERVER SESSIONS AND DYNAMIC RECONNECTION OF USERS TO PREVIOUS SESSIONS

TECHNICAL FIELD

This invention relates to client-server computer systems. More particularly, this invention relates to server operating systems executing on servers in the client-server computer systems.

BACKGROUND OF THE INVENTION

Most people are generally familiar with computer operating systems. An operating system is the code or computer program that provides an environment in which other computer programs can run. The operating system manages the system resources, such as processing resources, memory allocation, and I/O devices, to enable other programs to easily take advantage of these resources. Some operating systems are multitasking, meaning they support concurrent operation of more than one task. A multitasking operating system divides the resources among various processes, giving each process access to memory, processing resources, and at least one execution thread.

In the client-server environment, the server operating system is also called upon to handle communication interaction with client-based applications. The client operating system locates a server and requests a connection with the server and the server allocates resources to handle requests from the client.

One well-known server operating system is the Windows NT server operating system from Microsoft Corporation. The Windows NT server operating system is a multitasking, extensible, scalable operating system that can be implemented on standard personal computers. The Windows NT server operating system is well documented. For background information, the reader is directed to the book entitled *Inside Windows NT*,

With the WinFrame technology, a client can connect to the Windows NT server and begin a windowing session. To the user, it appears as if the client is a standalone Window-enabled computer that is running its own Windows-brand operating system. However, the session is actually taking place on the server computer, remote from the client, and the client is merely running a local graphical user interface to provide entry to the session. In this regard, the WinFrame technology is particularly well suited for low intelligent client ~~computers, such~~ as terminals and network-centric ~~computers, because~~ the client need not be capable of running its own operating system. However, the WinFrame technology is equally well suited for fully enabled clients.

One problem with the Citrix WinFrame technology is a lack of usability from the user perspective. Each session is bound to a particular client machine and does not consider the requirements of a user, who may in fact log on to the server from different physical machines.

To exemplify the problem, consider a common scenario. A user logs onto the Windows NT server from a client computer at the user's workplace. The workplace client is a desktop PC with a large 20" VGA display with 1024 x 768 resolution. At quitting time, the user decides to temporarily halt the session and disconnects from the server. Later that evening, the user reconnects to the server from a home computer, which is a laptop computer with an 11" color display of 640 x 480 resolution. When the laptop connects to the server, the Citrix WinFrame technology creates a new session for this new client machine. Since the user is employing different clients, the user is not permitted to reconnect to the old session. As a result, the user's previous session started on his/her work computer is not ported to the laptop computer because the server understands them as two distinct sessions started on two distinct machines.

Accordingly, there is a need to improve the server operating system to overcome this problem and to better focus on the needs of the user.

assigns a second protocol stack the second client-server connection. The session manager then begins creating a new, second session for the user.

During this process, however, the session manager recognizes that the user is affiliated with the first session. The session manager adapts the first session to conform to the system configuration of the second client. That is, the first session is reconfigured with a new set of one or more configuration parameters (e.g., a new display driver or protocol driver) to support the different system configuration of the second client.

The session manager then reassociates the second protocol stack with the reconfigured first session so that the user is returned to his/her original session, even though they logged on from a different client. The adaptation occurs solely at the server, and hence, it is invisible to the user. As a result, the user merely sees the same session. In the case of a changed display driver, for example, the user would see the same session, yet reformatted for the display of the second client.

Another aspect of this invention concerns using dynamic reassociation to accommodate multiple users who connect to the server from a single client computer. As an example of this situation, it is common for two bank tellers to concurrently use a single computer. In this case, the server detects when different users log onto the server via the same client. Upon detection, the server reassociates the protocol stack handling the client-server connection with the session affiliated with the present user. In this manner, the server can support multiple users, all of whom use the same computer.

BRIEF DESCRIPTION OF THE DRAWINGS

The same reference numbers are used throughout the drawings to reference like components and features.

Fig. 1 is an illustration of a client-server computer system.

Fig. 2 is a block diagram of a server computer in the client-server computer system.

The clients 24(1)-24(5) represent various kinds of computers that may connect to the server 22 over the network 26. Client 24(1) is a conventional desktop personal computer (PC), which possesses a local operating system, processing unit, and storage system. As one example, the desktop client 24(1) is a general-purpose PC implemented with a Windows-brand operating system from Microsoft Corporation, such as Windows 95 or Windows 98. The desktop client 24(1) is a standalone computer that primarily interfaces ~~with the server~~ to access files or other information ~~that is not~~ locally stored.

Client 24(2) is illustrated as a portable laptop computer, which can be connected to the network 26 via a conventional network card or a dial-up modem connection. The laptop computer 24(2) is also a fully equipped, standalone computer that can be configured with its own operating system, processing unit, and storage system. The laptop client 24(2) may likewise be configured to run a Windows-brand operating system, such as Windows 95 or Windows 98.

Client 24(3) is a handheld PC, which generally possesses less functionality than a general-purpose computer. However, the handheld PC 24(3) is preferably equipped with a Windows-brand operating system, such as Windows CE from Microsoft Corporation.

Client 24(4) is a terminal computer, which is a low cost machine with minimal local processing and little local memory. The terminal computer includes a display, a keyboard, a mouse (optional), and enough intelligence to connect to a server. All applications run at the server and the terminal merely provides a connection point to the server-based processing.

Client 24(5) represents a network-centric computer, such as a Network Computer (or NC) or a Net PC. The network-centric computer client 24(5) primarily facilitates execution of server-based applications, but has some limited local processing capabilities for running Windows-based or Java-based applications.

60, one or more application programs 62, other program modules 64, and program data 66.

A user may enter commands and information into the server 22 through input devices such as a keyboard 68 and a mouse 70. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 32 through a serial port interface 72 that is coupled to the ~~system bus 36,~~ but may alternatively be ~~connected by other~~ interfaces, such as a parallel port, game port, or a universal serial bus (USB).

A monitor 74 or other type of display device is also connected to the system bus 36 via an interface, such as a video adapter 76. The server computer 22 has a network interface or adapter 78, a modem 79 or other means for establishing communications over the network 26, such as a T1 or T3 Internet connection. The modem 79 facilitates dialup connection from a client.

General Software Architecture

The computer system 20 implements a software architecture that enables the clients 24(1)-24(5) to run a Windows-brand operating system and applications on the server 22. The software architecture includes a server-based multi-client core, a client-based module, and a remote protocol that enables communications between the server and client.

The server-based multi-client core enables the server to host multiple, simultaneous client sessions. The multi-client core is resident as part of the operating system 60 and executes on processor 32. The multi-client core supports both windowing-enabled clients (i.e., clients that are capable of running a local windowing operating system, such as the desktop client 24(1), the laptop client 24(2), and the handheld PC client 24(3)) and non-windowing clients (i.e., clients that are incapable of running a local

In contrast to prior art client-server systems, the computer system 20 is tailored to view the server network as servicing the needs of a user, rather than supporting specific client computers. This ideology shift implies that the computer system 20 supports the user, regardless of which client 24 the user employs to access the server 22. Prior art client-server systems typically emphasize a machine-centric perspective, treating each connection in terms of the needs of a particular client configuration.

As an example of this dichotomy, a scenario presented in the Background section describes a user who logs onto the server from a workplace client computer, begins a session, stops the session without terminating it, and subsequently logs on from a home computer. In the prior art architecture, the server creates a new session for each client because they originated from two different clients with different system configurations. When the user logs on from the first client, the server creates a session that utilizes display drivers, keyboard drivers, and the like that are specific to support the particular client computer. The server might also use one type of protocol that is suitable for communicating with the client computer.

When the user logs on from the second client, the server creates another session that loads different display drivers, keyboard drivers, and the like that are specific to support the second client computer. The server might also employ a different type of communication protocol. Hence, the server creates machine-dependent sessions, even though the same user would like to continue the same session regardless of the machine from which he/she uses to connect to the server. The prior art represents a machine-centric architecture.

The computer system 20, on the other hand, is user centric. In the above scenario, the computer system 20 enables the user to connect to the server from one client, start and stop a session without termination, and later return to that same session from a different client. The server does not create a new session for the new client, but instead adapts the

any new client system configurations. This dynamic reassociation takes place entirely at the server.

Fig. 3 shows a portion of the server operating system 60, including the multi-client core that enables replication of the windowing modules to form multiple windowing sessions. The operating system 60 includes code that runs in a privileged processor mode, which is often referred to as the "kernel mode", and applications or other service routines that ~~run in a non-privileged processor mode, commonly referred to as~~ "user mode".

The operating system 60 includes a multi-user session manager 90 implemented in the user mode. The multi-user session manager 90 enables creation of multiple client-server windowing sessions that reside on the server for use by the client. The operating system 60 further includes a protocol stack manager 92 implemented in the kernel mode and closely interfaced with the session manager 90. Under the direction of the multi-user session manager, the protocol stack manager 92 creates and manages one or more protocol stacks, as represented in Fig. 3 by protocol stack 94, to facilitate communication with corresponding external clients during the client-server windowing sessions. The protocol stacks are implemented according to certain types of communications protocols, such as RDP over a TCP/IP transport, RDP over IPX, RDP on top of an asynchronous serial connection (modem), and the like.

One exemplary implementation of the session manager 90 and the protocol stack manager 92 is using the Intelligent Console Architecture (ICA) from Citrix Systems, Inc. The Citrix ICA extends the Windows NT server operating system to provide multiple windowing sessions for connected clients.

For discussion purposes, suppose a client 24(1) desires to run a server-based windowing session. The client 24(1) is illustrated as a Windows-enabled desktop computer. However, other types of clients may be used, including terminal clients and network-centric clients. The client 24(1) sends a connection request to a well-known

collection of hardware device drivers for the hardware display, the GDI interface to the hardware display (GDI driver), the mouse driver, and keyboard driver.

In new versions of the Windows NT server operating system such as Windows NT 4.0 or 5.0, the USER and GDI portions of the CSRSS module 98 are moved to the kernel mode, to a module referred to as WIN32K.SYS 100. The CSRSS module 98 and Win32K module 100 are closely integrated across the user-kernel line. Fig. 3 shows the software modules implemented in the newer versions of Windows NT.

When a client requests a windowing session, the server operating system creates the CSRSS module 98 and Win32K module 100 to open the hardware devices and load the proper GDI display driver 102 and protocol driver 104 that are compatible with the remote client's system configuration. In this example, the client 24(1) is a desktop computer that has a VGA monitor with a resolution of 1024 x 768 and uses a RDP protocol. After these drivers are loaded, the protocol stack 94 is bound to the windowing session 1. Communications with the client are passed to the windowing session 1, as illustrated by the arrow from the protocol stack 94 to the protocol driver 104.

As part of the session startup, the user at the remote client 24(1) is prompted via a Windows logon (WL) module 106 to enter a user identification (ID). The operating system associates the user ID with the windowing session 1. This is represented by the box labeled "User 1" in the user mode for the windowing session 1.

The server operating system then provides the windowing environment on the client computer 24(1). The client-based code runs a windowing graphical user interface that enables the user to start applications on the server, and perform other functions that are common to Windows-based systems.

With the first protocol stack 94 bound to session 1, the server operating system 60 prepares to accommodate another client by creating a container for a second protocol stack. When a second client connects to the server to run a windowing session, the stack manager 92 assigns the protocol stack to the new client.

windowing session from one client computer and logs onto the server using another client computer, the WinFrame technology will create a new windowing session in the case where the two client computers do not share common video resolutions. The user is not permitted to reconnect to the old session.

In the client-server system of this invention, however, the server operating system is designed to accommodate the user's needs and to support windowing sessions that a user ~~can reconnect to~~ from different machines. Recall ~~from the above example~~ that user 1 is connected to the server using the desktop client 24(1) with a 1024 x 768 VGA monitor and a RDP protocol.

To continue this example, suppose that user 1 disconnects from the server 60, without terminating the windowing session 1. The user leaves the workplace and goes home. While at home, the user decides to continue the session that he/she began earlier in the day. The user connects to the server from a second client computer located at home.

Fig. 5 shows the situation in which user 1 first connected to the server using a first client 24(1) to create a first client-server windowing session 1, as described above with respect to Fig. 3, and subsequently connects to the server using a second client 24(2). Upon disconnection of the first session, the container previously loaded with RDP and TCP/IP stack drivers is emptied and its associated communications endpoint is destroyed. This is represented by the empty protocol stack 94(1).

When the user subsequently logs on using the second client 24(2), the multi-user session manager 90 initially creates a second session 2, as if the person logging on is a new user. In this example, the client 24(2) is a laptop computer with a flat panel, 11" color display with a resolution of 640 x 480. The client 24(2) uses a RDP protocol. Accordingly, the second client 24(2) has a different system configuration than the first client 24(1) used in Fig. 3.

session 1. In the preferred implementation, the stack contents (the WinStation Driver (WD), protocol driver (PD), and transport driver (TD)) from the second stack and the live connection endpoint are moved back over to the first protocol stack 94(1) for the previous session 1.

Fig. 6 shows the results of modifying the Win32K module 100(1) to support the new client computer. The display driver 102(1) is now a driver for the 640 x 480 flat panel color display. Additionally, the stack contents are loaded back into the protocol stack 94(1) for session 1. As a result, the server dynamically adjusts to the client computer that the user is now employing as an access point. Session 2 and its associated protocol stack 94(2) will be destroyed once the re-association is complete.

In this example, the display data is dynamically resized to fit the user's new client computer 24(1). The session appears the same to the user, only it is reformatted for the user's smaller, lower resolution screen. The previous WinFrame technology did not permit this reassociation process, and hence the original session could not be served to the new client computer 24(1).

The above example highlights adapting the server session to conform to a different display size and resolution. This is merely one example. Other possible client configuration parameters that may be accounted for at the server include types of keyboards, keyboard layouts (QWRTY v. Dvorak), protocol types (e.g., RDP over TCP/IP, versus RDP over asynchronous modem), language (e.g., English v. Spanish), memory drives, and peripherals.

Fig. 7 shows the steps in the dynamic reassociation process. These steps are implemented in code within the server operating system, preferably as part of the multi-user session manager with user authentication software in the Windows logon module. At step 120, a user connects to the server via a client. The server commences the connection process to form a client-server connection (step 122) and begins creating a new client-server windowing session (step 124).

operating system associates the protocol stack 94(1) with session 1 as represented by the solid arrow 140.

When user 1 completes a task, the user freezes the session without logging off or otherwise terminating the session. Another user is then free to use the terminal 24(4). Suppose user 2 logs onto the server. The session manager 90 creates a second session 2 for the second user 2. The Win32K module in session 2 also provides the appropriate drivers and devices to implement a windowing session on the terminal 24(4). The operating system associates the second protocol stack 94(2) with session 2 as represented by the dashed arrow 142. This process can continue for the N users.

Now, when the terminal is again free to use, user 1 may wish to return to session 1. The user retypes his/her ID. The operating system initially begins to create a new session and associated protocol stack. However, when the operating system realizes that the user has a pre-existing disconnected session, the operating system reassociates the user with the existing session 1. The stack contents and live communications endpoint are transferred back to the protocol stack container 94(1) for association with the session 1. In this manner, the server operating system accommodates multiple user sessions from the same client computer.

Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

4. An operating system embodied on a computer-readable medium having computer-executable instructions for performing the computer-implemented method as recited in claim 1.

5. A server computer programmed to perform the computer-implemented method as recited in claim 1.

6. In a client-server computer system, a computer-implemented method implemented at a server comprising the following steps:

creating a connection with a first client, which is being operated by a user;

assigning a first protocol stack to the first client;

logging on the user;

establishing a first session for the first client;

configuring the first session to conform to configuration parameters of the first client;

associating the first session with the first user;

ending the connection with the first client;

subsequently creating a connection to a second client, which is being operated by the same user;

assigning a second protocol stack to the second client;

logging on the user;

detecting the user as being associated with the first session;

reconfiguring the first session to configuration parameters for the second client; and

reassociating the first session with the user by transferring contents of the second protocol stack to the first protocol stack associated with the first session so that the user may continue with the first session while using the second client.

11. An operating system embodied on a computer-readable medium having computer-executable instructions for performing the computer-implemented method as recited in claim 10.

12. A server computer programmed to perform the computer-implemented method as recited in claim 10.

13. In a client-server computer system in which a client is connected to communicate with the server via a client-server connection, a computer-implemented method implemented at a server comprising the following steps:

creating a first session for a first user who accesses to the server from the client;

associating the client-server connection with the first session;

temporarily freezing the first session;

creating a second session for a second user who connects to the server from the client;

associating the client-server connection with the second session;

temporarily freezing the second session; and

subsequently reassociating the client-server connection with the first session.

14. An operating system embodied on a computer-readable medium having computer-executable instructions for performing the computer-implemented method as recited in claim 13.

15. A server computer programmed to perform the computer-implemented method as recited in claim 13.

20. An operating system as recited in claim 16, wherein the system configuration of the first and second clients include parameters selected from a group of parameters comprising display type, keyboard type, network protocol type, peripheral types, human language, and memory drives.

21. An operating system for a server computer, wherein the operating system is embodied ~~on a computer~~-readable medium, comprising:

a multi-user session manager to enable multiple client-server sessions on the server;

a multi-user stack protocol manager to manage one or more protocol stacks, the protocol stacks facilitating communication with corresponding clients during the client-server sessions;

in an event that a user connects to the server from a first client, the stack protocol manager assigns a first protocol stack to a first client-server connection and the session manager creates a first session for the user, the first session being configured to accommodate a system configuration of the first client;

in an event that the user exits the current session and subsequently reconnects to the server from a second client different from the first client, the stack manager assigns a second protocol stack to a second client-server connection;

wherein the session manager recognizes the user as being affiliated with the first session and adapts the first session to conform to a system configuration of the second client, the session manager transferring contents of the second protocol stack to the first protocol stack associated with the first session

25. In a windows-based server operating system embodied on a computer-readable medium at a server, wherein the server operating system is capable of providing multiple client-server graphical user interface windowing sessions for multiple clients, an improvement to the server operating system comprising:

code means for maintaining sessions at the server after clients disconnect from the server;

code means for detecting whether a user, who reconnects to the server through a new client, is already affiliated with a previous session;

code means for adapting the previous session to conform to a system configuration of the new client in an event that the system configuration is different from that of a previous client through which the user was previously connected to the server; and

code means for reassociating the user with the previous session.

26. In a windows-based server operating system embodied on a computer-readable medium at a server, wherein the server operating system is capable of facilitating multiple client-server sessions in which the server provides a windowing environment for a client, an improvement to the server operating system comprising:

computer readable instructions to direct a server to recognize when a user, who is connecting to the server via a remote client, is associated with a previous windowing session that the user started previously;

computer readable instructions to direct the server to adapt the previous windowing session to support a new display type in the event that the new display type is different than an original display type on a client from which the user originally connected to the server to start the previous windowing session; and

computer readable instructions to direct the server to reassociate the user with the previous windowing session and to reformat data served from the previous windowing session for depiction on the new display type.

windowing session, the server-based operating system adapting the previous windowing session to conform to any different system configurations of the client in the event that the client is different than a client from which the user originally connected to the server to start the previous windowing session.

29. A system as recited in claim 28, wherein the client-based code is configured to run on a client that is implemented with an operating system that is capable of connecting to the server.

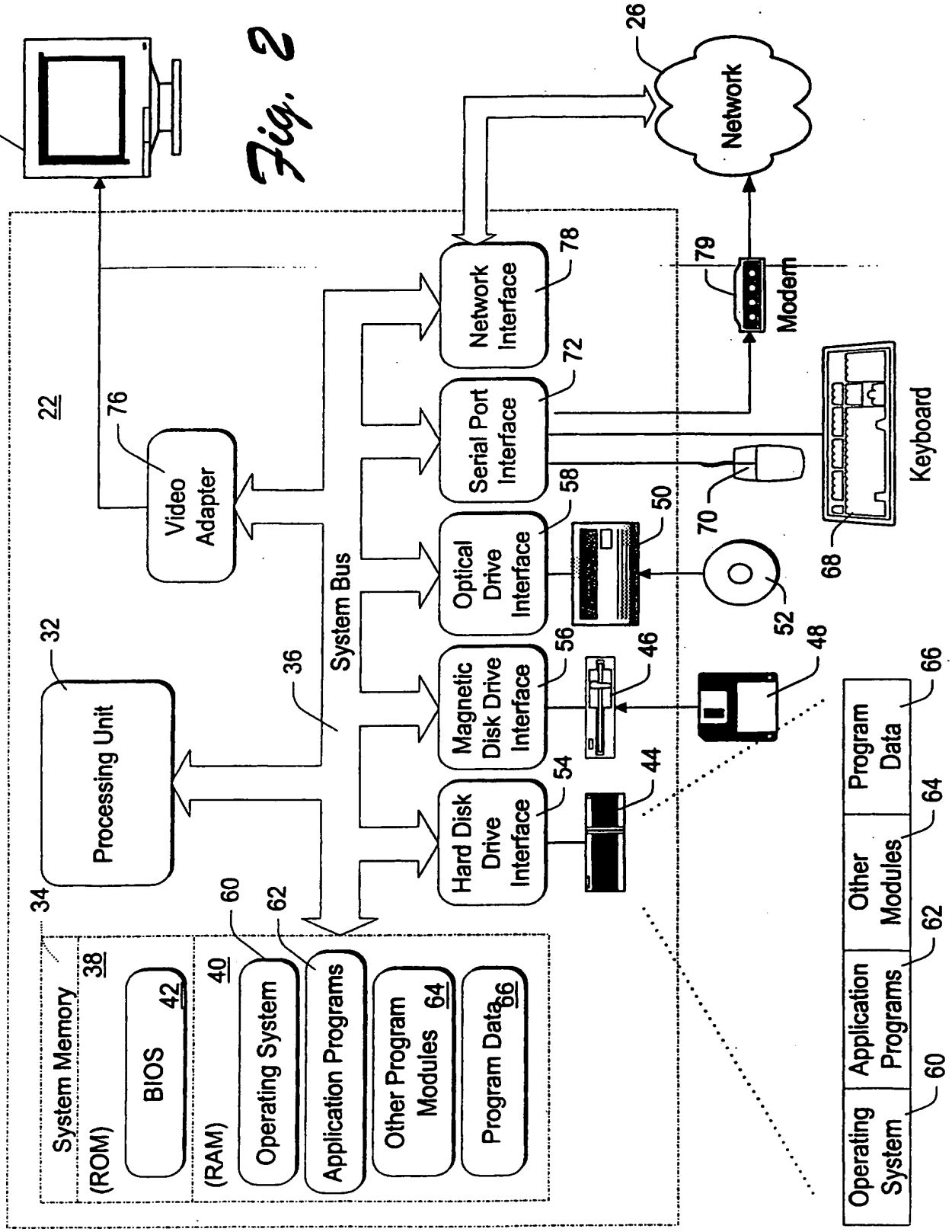
30. In a client-server computer system, a system comprising:

a server-based operating system embodied on a computer-readable medium and executed at a server, the server-based operating system enabling multiple graphical user interface windowing sessions for multiple clients;

a client-based code embodied on a computer-readable medium and executed at a client, the client-based code enabling the client to initiate a connection with the server and to run a local graphical window user interface (UI) to the server-based windowing sessions; and

the server-based operating system being configured to associate communication to and from the client-based code with various ones of the windowing sessions so that multiple users can employ the client to operate their own corresponding windowing sessions.

31. A system as recited in claim 30, wherein the client-based code is configured to run on a client that is implemented with an operating system that is capable of connecting to the server.


74
Fig. 2
26
Network
22
76
Video Adapter
Network Interface
78
79
Modem
Serial Port Interface
72
Keyboard
70
68
Optical Drive Interface
58
50
52
System Bus
Magnetic Disk Drive Interface
56
46
48
32
Processing Unit
36
Hard Disk Drive Interface
54
44
34
System Memory
38
(ROM)
BIOS
42
40
(RAM)
60
Operating System
62
Application Programs
Other Program Modules
64
Program Data
66
Operating System
Application Programs
Other Modules
Program Data

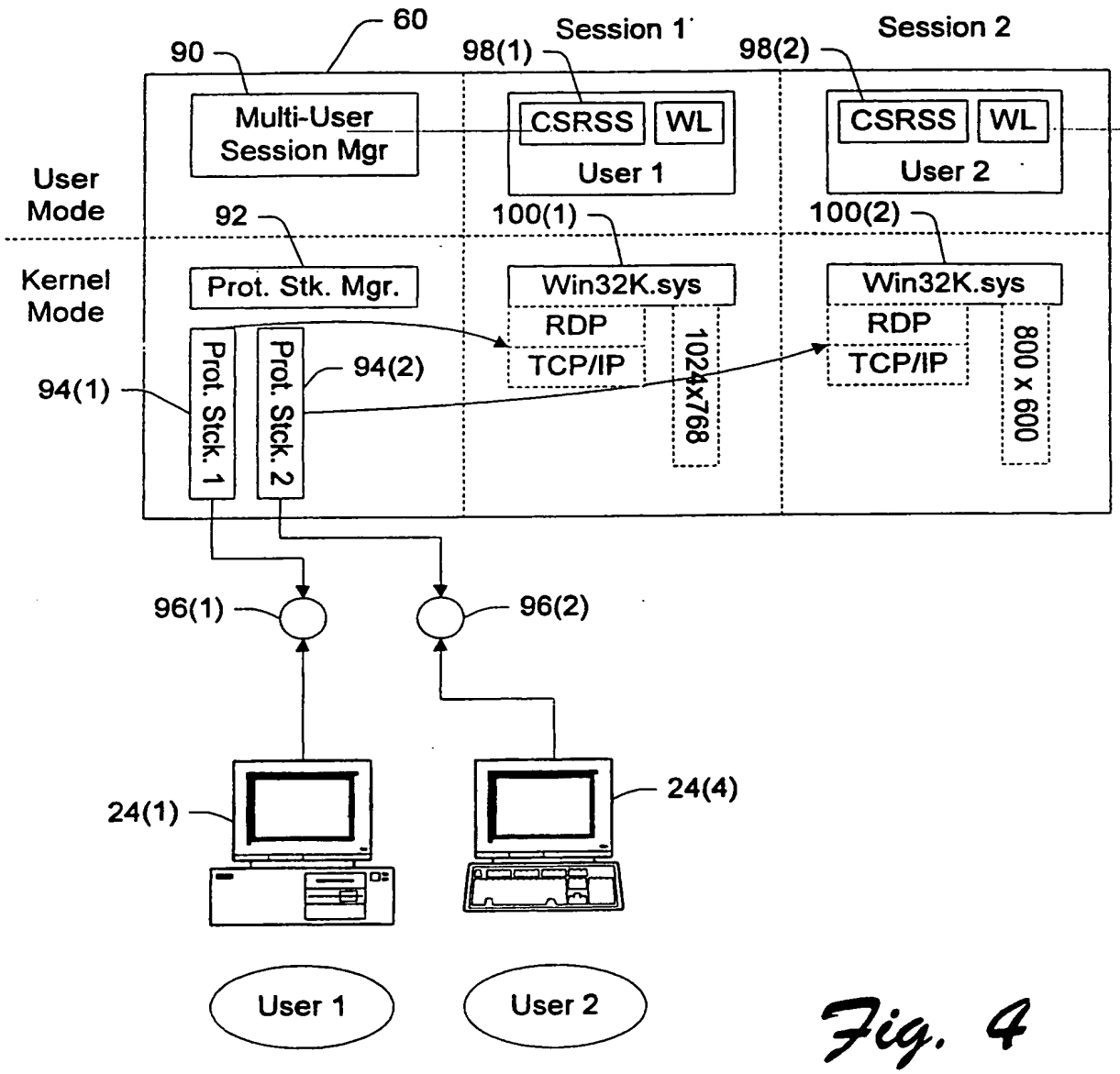

60
90
98(1)
Session 1
98(2)
Session 2
Multi-User Session Mgr
CSRSS
WL
User 1
CSRSS
WL
User 2
User Mode
92
100(1)
100(2)
Kernel Mode
Prot. Stk. Mgr.
Win32K.sys
RDP
TCP/IP
1024x768
Win32K.sys
RDP
TCP/IP
800 x 600
94(1)
Prot. Stck. 1
Prot. Stck. 2
94(2)
96(1)
96(2)
24(1)
24(4)
User 1
User 2

Fig. 4

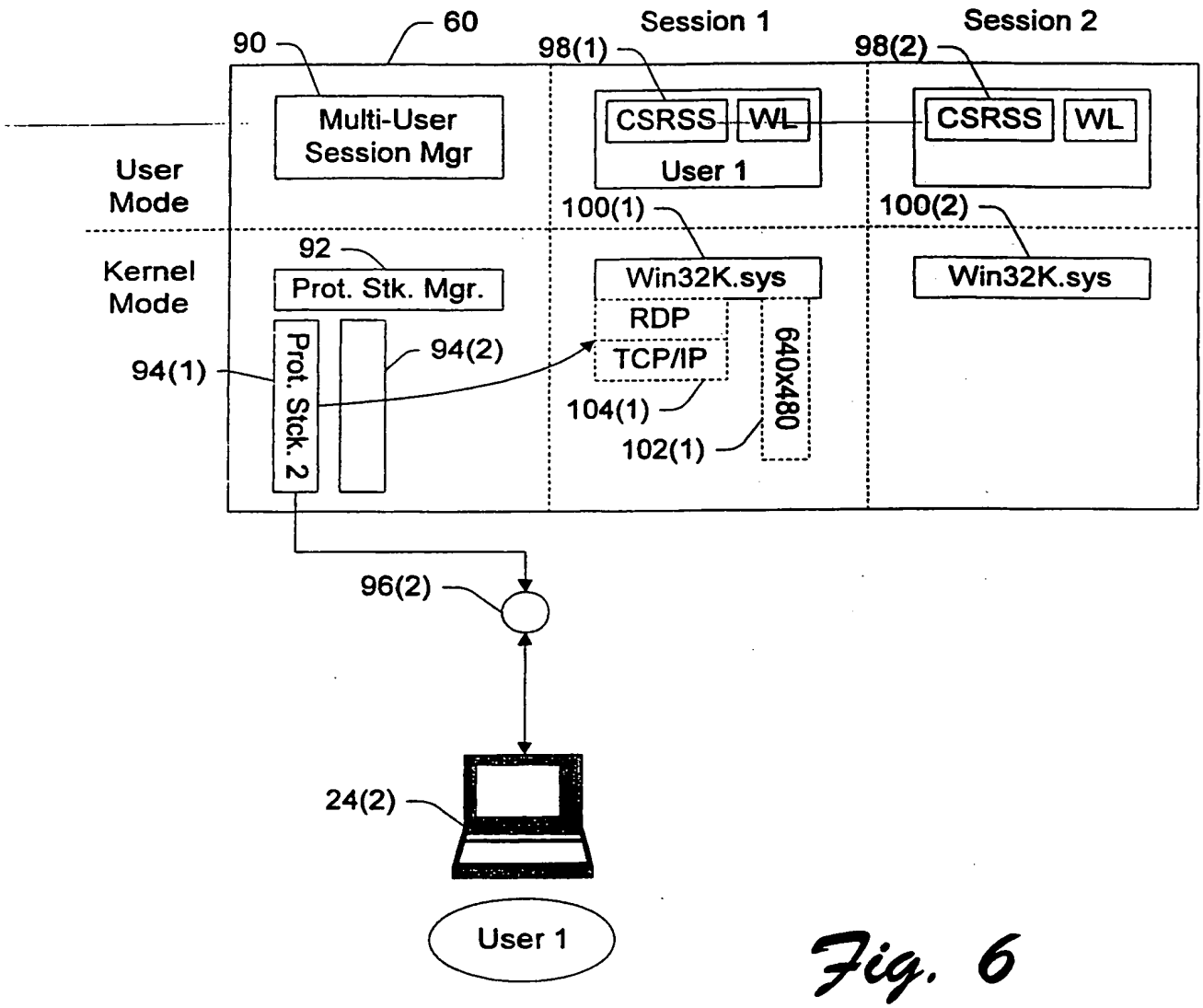

60
Session 1
Session 2
90
98(1)
98(2)
Multi-User
Session Mgr
CSRSS
WL
CSRSS
WL
User 1
User
Mode
100(1)
100(2)
92
Kernel
Mode
Prot. Stk. Mgr.
Win32K.sys
Win32K.sys
RDP
TCP/IP
640x480
94(1)
94(2)
Prot. Stck. 2
104(1)
102(1)
96(2)
24(2)
User 1

Fig. 6

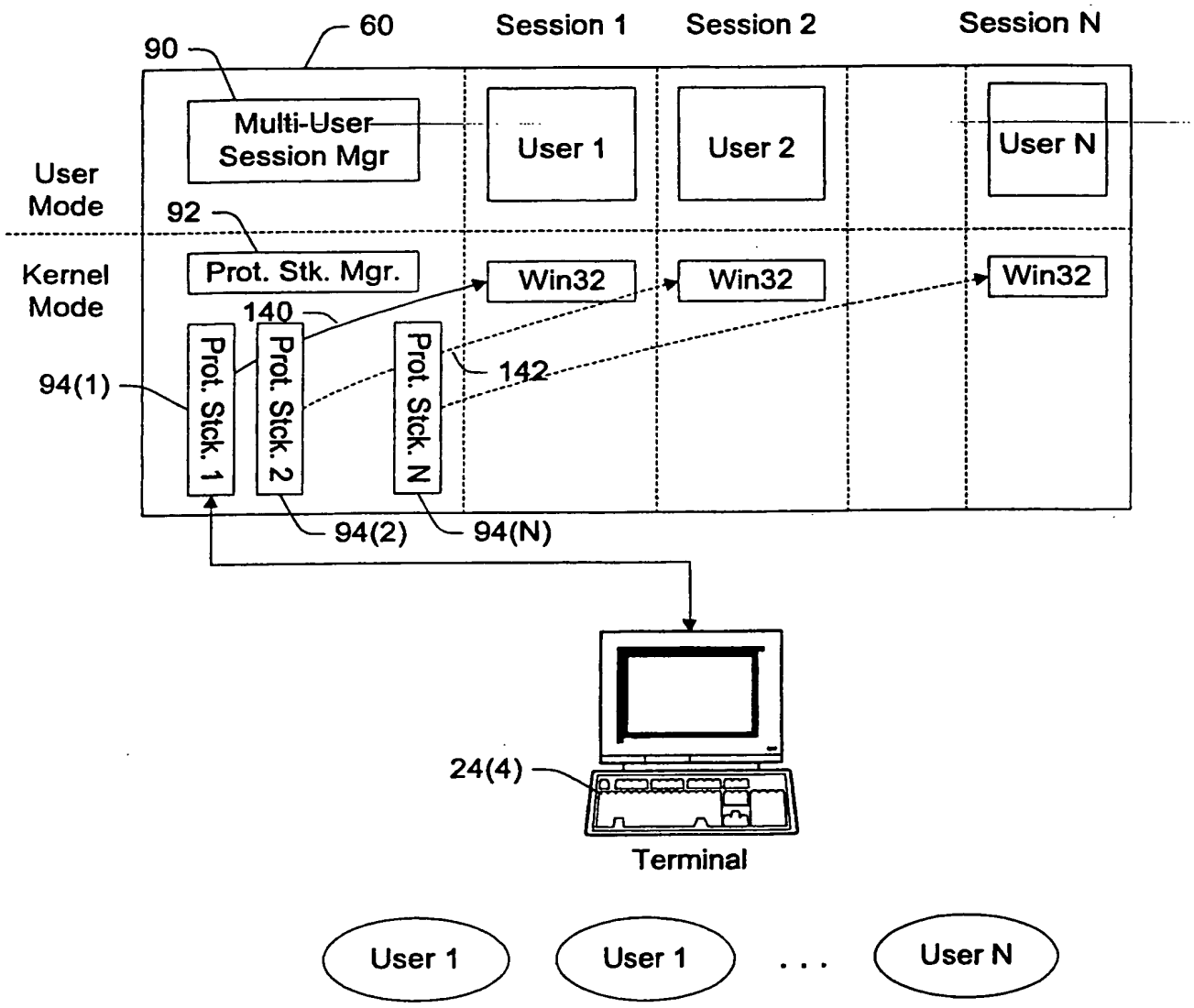

60
Session 1
Session 2
Session N
90
Multi-User Session Mgr
User 1
User 2
User N
User Mode
92
Kernel Mode
Prot. Stk. Mgr.
Win32
Win32
Win32
140
94(1)
Prot. Stck. 1
Prot. Stck. 2
Prot. Stck. N
142
94(2)
94(N)
24(4)
Terminal
User 1
User 1
. . .
User N

Fig. 8